Tessella Project Number: 7596

**Tessella plc** 26 The Quadrant, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YS, UK

T: +44 (0)1235 555511 | F: +44 (0)1235 553301 | E: info@tessella.com | www.tessella.com

Client: RAL & ORNL

Mantid

CLI for Plotting



Document Control

This document is under document control. All members of the distribution list will receive updated copies whenever alterations are made. All other copies are uncontrolled.

Document Owner

|  |  |  |
| --- | --- | --- |
| Name | Role | Company |
| Owen Arnold | Owner | Tessella |

Document Approval

The following are required to approve this document. Their approval signifies that the document is fit for purpose and approved for use. .

|  |  |  |
| --- | --- | --- |
| Name | Approval Role | Company |
|  |  |  |

Distribution

|  |  |  |
| --- | --- | --- |
| Name | Role | Company |
| Mantid GIT repository | Filing | GitHub |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Issue | Date | Description |
| Owen Arnold | V1.R1.M0 | 25th-February-2014 | 1st version written |

References

|  |  |  |  |
| --- | --- | --- | --- |
| Document | Ref | Date | Details & Issue |
| Scientific Steering Committee meeting minutes 2014 | <http://www.mantidproject.org/SSC_2014_Mantid_General> |  |  |

Table of Contents

1 Introduction 3

2 Objectives: 4

2.1 Detailed Objectives 4

3 High-level Proposed Solution(s) 4

3.1 Current Types 4

3.2 Current Example Usage 5

3.3 Prototype example usage 5

4 Outstanding Questions 5

# Introduction

Purpose of this Document: This document describes the detailed design of the new CLI for python control of plotting in the MantidPlot application.

# Objectives:

The Scientific Steering meeting of 2014 highlighted some major issues around the current plotting in MantidPlot. Some of these issues relate to the command line interface (CLI). A full listing of the points can be found here: <http://www.mantidproject.org/SSC_2014_Mantid_General>. The objective will be to provide a better CLI following the advice and comments provided by instrument scientists.

## Detailed Objectives

* Want simpler control over the plot options
* Want a limited number of plot options. For the default new CLI configuration.
* Want the ability to switch plotting style/implementation. Ensure that no changes are made that would impede us from support more than one plotting interface. For example Horace style plotting.
* We should maintain backwards compatibility with current qtiplot as many interfaces will break if not.
* Additional top-level **plot** command that will inspect the data and plot it in the most sensible form.
* Plot should take an optional tool input, to select which plotting tool to use.
* Overwrite **plot{...}** variants to return a user-friendly plot object
* **plot{...}** variants to have a common set of controls, such as for setting log scale, and controlling individual graphs
* Overwrite **plot{...}** variants to return a user-friendly plot object
* Add **plotInstrument** option
* Consider python control for future interface exposure such as VSI and tile view.

# High-level Proposed Solution(s)

* Additional top-level plot command that will inspect the data and plot it in the most sensible form.
* Wrap and extend existing qtiplot python functionality in all cases. Existing behaviour must be preserved until it can be phased out.
* Use forwarding methods expose the activeLayer functionality to the returned plot handle (MultiLayerPlot proxy)
* Possible to have a new 'Facade' type to expose all options. This could be a new type (returned by the plot methods), which encompasses MultiLayerPlot, Graph and Legend, as well as, ErrorBarSettings etc. It would avoid the need for users to drill-down and fetch the relevant objects to access aspects of the control.

## Current Example Usage

plot\_handle = plotSpectrum(source=[{Workspaces}], indices=[{Indexes}])

graph = plot\_handle.activeLayer()

graph.logLogAxes()

## Prototype example usage

### Plot

Plot(source={Workspace}, tool={ToolName}, \*\*kwargs)

Return: Easiest thing to do would be to return whatever it is that the individual tool returns. More complex, but possibly more useful thing to do would be to return some kind of abstraction which would give access to common utilities on all tools.

### Plot Spectrum

plot\_handle = plotSpectrum(source=[{Workspaces}], indices=[{Indexes}])

plot\_handle.logLogAxes()

Return: current MultiLayer type object (with Graph forwarding methods) or Façade.

Expose common options such as log axis and line colours as function arguments

plotSpectrum([{Workspaces}], [{Indexes}], Axes='LogLogAxes')

### Plot Instrument

instrument\_view = plotInstrument({Workspace})

render\_tab = instrument\_view.getInstrumentTab()

## Current Types

<https://github.com/mantidproject/mantid/blob/master/Code/Mantid/MantidPlot/src/Graph.h>

<https://github.com/mantidproject/mantid/blob/master/Code/Mantid/MantidPlot/src/MultiLayer.h>

# Detailed Solution

TODO

# Outstanding Questions

* Should **plotSpectrum** have the option to actually plot spectrum numbers rather than workspace indexes?
* Do we want **plotInstrument** and **plotVSI**?
* Module names and structure. Ability to change default in preferences.?
* An automated way of checking that the existing functionality hasn’t been negatively affected would be useful.